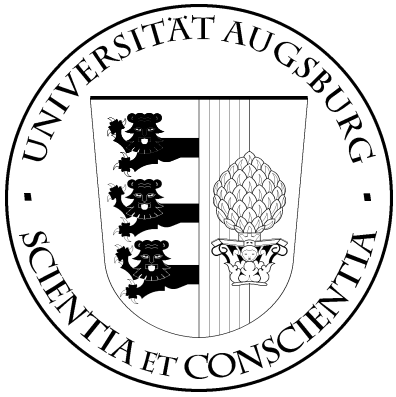
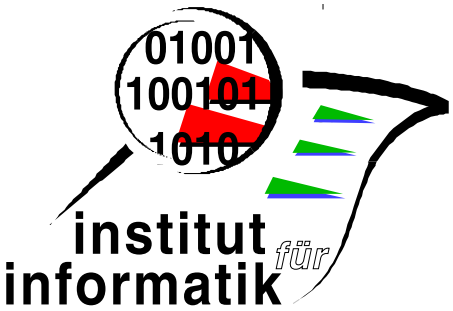
Universität Augsburg



Good, Better, and Most Probable Recommendations

Martin E. Müller

Report 17 2004



Institut für Informatik
D-86135 Augsburg

Note.

This technical report is a revised version of earlier material which partially has served as a basis for the later publications [Müller 2004b] and [Müller 2004a].
Accordingly, due to the common base, some parts are similar; however, both above mentioned version lack a closer description of Bayesian inference and information gain methods. [Müller 2004b] focuses on *theoretical aspects* and includes quite a big deal on *planning*, while [Müller 2004a] focuses on applications in the context of *adaptive hypermedia*.
This article mainly focuses on *applications* of machine learning techniques in *recommender systems*; hence the title.

Good, Better and Most Probable Recommendations

MARTIN E. MÜLLER
University of Augsburg

Machine Learning seems to offer the solution to the central problem in recommender systems: Learning to recommend interesting items from observations. However, one tends to run into similar problems each time one tries to apply out–of–the–box solutions from ML.

This article relates the problem of recommendation by user modeling closely to the machine learning problem and explicates some inherent dilemmas. A few examples will illustrate specific approaches and discuss underlying assumptions on the domain or how learned hypotheses relate to requirements on the user model. The article concludes with a tentative 'checklist' that one might like to consider when thinking about to use ML4UM techniques.

Categories and Subject Descriptors: H.4.0 [**Information Systems Applications**]: General; H.1.2 [**User/Machine Systems**]: User Modeling; I.2.6 [**Learning**]: Machine Learning for User Modeling

Additional Key Words and Phrases: Machine Learning for User Modeling, Recommender Systems, Adaptive User Interfaces

1. INTRODUCTION

In [Webb et al. 2001], the authors point out several problems that are likely to be overseen when trying to learn user models. It seems, that:

> *At first blush, user modeling appears to be a prime candidate for straightforward application of standard machine learning techniques.*

This apparently obvious observation describes blooming research on machine learning for user modeling, where one popular application are recommender systems.

However, the application of a variety of machine learning algorithms implies certain restrictions on the domain—and issues of Human–Computer interaction have certain implications on the availability of data for learning. This article presents a detailed description of what both machine learning and user modeling require from and what they can deliver to each other. By relating the general user modeling problem to the general machine learning problem several hidden problems will become more explicit and thus, avoidable.

Author's address: Martin E. Müller, University of Augsburg, Computer Science, Multimedia and Applications, Eichleitnerstr. 30, 86159 Augsburg, Germany.

1.1 Learning about the user

Adaptive user interfaces adapt themselves to the user by reasoning about the user and refining their internal model of the user's needs. The refinement of the user model is a sequence of inferences based upon several observations of user interactions, commonly known as feedback (already introduced into the information retrieval community in [van Rijsbergen 1979]). This yields an abstract architecture as shown in figure 1. A user's information *request* is translated into a system

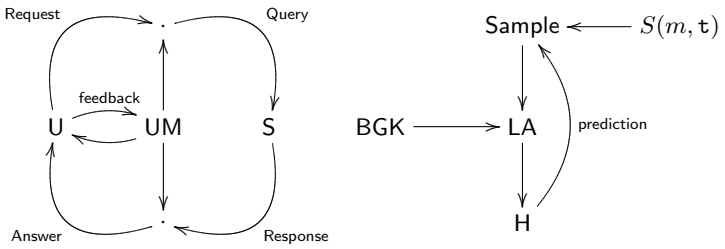


Fig. 1. An abstract AUI and an abstract ML system

query by taking into account knowledge about the domain (i.e. the system S) and knowledge about the user (taken from the internal user model UM). The system's *response* is transformed (by filtering, arrangement, aggregation etc.) again taking into account the user model UM into an *answer* (containing recommendations) that is presented to the user. The user interactions (including explicit *feedback*) are used to refine the user model in order to be able to deliver better results next time (i.e. more accurate recommendations).

In machine learning (ML), artificial systems learn how to perform better by experience. By observing examples from a *sample*, the learning algorithm LA tries to induce a hypothesis H which approximates a general, unknown law that explains the nature of all objects of the domain (both observed and unknown). This inductive inference is supported by a set of background knowledge BGK. As long as the hypothesis and according predictions disagree with observations, the hypothesis needs to be refined. This process is depicted at the right hand side of figure 1.

It seems obvious, that machine learning can offer what is needed as a feedback loop in user adaptive systems.

1.2 The problem remains the same

However, when trying to adapt by learning one will sooner or later encounter one or more well known problems (some of which have been discussed in [Webb et al. 2001]):

(1) There is only very little feedback available such that learning needs to yield results from very few examples only.; sometimes from positive evidence only. This is due to the fact that a user in general is not willing to give feedback as this requires additional effort and a significant benefit cannot neccessarily observed immediately.

(2) Learning of content based user models requires a sophisticated domain theory and thus, a lot of background knowledge while learning in collaborative user modeling sometimes results in strange user models.
It is clear, that domain knowledge has to be put into the machine by some kind of a knowledge engineering process. While individual user modeling requres a content based user model in terms of a domain description, collaborative models consist simply of sets of objects which are much easier to derive. But on the other hand, they cannot be explained to the user in a comparable scrutable way as individual, content based models. This may result in a loss of trustworthiness because humans need at least to feel to be in charge and this requires the system to be understandable to the user.

(3) Learning in presence of very large datasets takes a reasonable amount of time which is undesirable in many applications where a quick adaption is needed.
It is clear, that more data and more complex knowldege implies greater complexity. But on the other hand, human users are, in general, impatient. This means, that a latency should be kept as small as possible

Recently it seems that emphasis was put on the latter issue for two reasons: First, growing interest of industry for applicable methods which promise an advantage on the market requires quick adaption and fast results. Second, growing computational power and standards for markup languages which allow to represent and interchange meta data might lead to the conclusion, that many problems our now within reach of tractability.

The overall dilemma remains: There seems to be no system which learns quickly, highly accurate, nearly domain independent, from few examples with literally no bias and delivers a user model that is understandable and contains breaking news about the user's characteristics. Each single problem favors a certain learning approach.

2. UNDERSTANDING USER MODELING AS A MACHINE LEARNING PROBLEM

It takes a reasonable amount of intelligence to learn about the environment. In general, learning is characterized as a generalization process which enables the learner to induce intensional knowledge (like rules or concepts) from examples. The newly acquired knowledge needs to explain observations from the past and shall also help to recognize new observations. In other words, an initial model is refined by several observations such that the new theory is modified in way which includes all known pieces of evidence as well as many as possible unseen examples (and, at the same time, excluding as many as possible counterexamples).

2.1 What makes a learning algorithm?

Let Ω denote the *Universe*, that is, the set of all objects of the domain. Any subset of Ω may form a *concept*. The goal of a learning algorithm $\mathbf{A}$ is to learn a hypothesis h which accurately describes a (unknown) *target concept* $c_{\mathbf{t}}$.

The target is characterized by a function $\mathbf{t}$ which (in the ideal case) delivers a noise–free binary classification on any object $x \in \Omega$: $x \in c_{\mathbf{t}}$ if and only if $\mathbf{t}(x) = 1$. A *sample* is a sequence of *examples*; and an example is an object of our domain together with its label as assigned by $\mathbf{t}$. Accordingly, a sample consisting of m

examples is a sequence delivered by a sampling function

$$S_\Delta(m, \mathbf{t}) = \mathbf{s} = [\langle x_1, \mathbf{t}(x_1)\rangle, \langle x_2, \mathbf{t}(x_3)\rangle, \ldots, \langle x_m, \mathbf{t}(x_m)\rangle]\,. \tag{1}$$

The problem is, that probability of occurrence of objects of the domain is not uniformly distributed. S_Δ can be regarded to as a biased choice function which draws objects x from Ω depending on Δ. $\mathbf{A}$ is expected to induce a hypothesis which allows to predict the next object without knowing Δ. It is obvious, that any knowledge or assumption about Ω is desperately needed in order to be able to learn some h which is sufficiently accurate with as few examples as possible.

To give a measure of the quality of $h = \mathbf{A}(\mathbf{s})$, one can (given Δ) compute the *error* of h. Basically, the error is the Δ–weighted sum of all those $x \in \Omega$ for which $h(x) \neq \mathbf{t}(x)$. If the error $e(h)$ does not exceed a boundary of ε, h is said to be ε*–good*.[1]

It is obvious, that one wants to minimize $e(h)$. Since Δ is unknown, one needs to give upper bounds for expected errors. Accordingly, 'learnability' means, that there is some $\mathbf{A}$ which on a sample $S(m, \mathbf{t})$ induces some h such that $e(h) < \varepsilon$ can be guaranteed with a confidence of $1 - \delta$. This may sound a bit difficult, but it is the strongest proposition one can show—since Δ is unknown.

The learning model described so far is the PAC learning model (see [Valiant 1984] and [Kearns 1990; Kearns and Vazirani 1994]). Its disadvantage is, that PAC learnability is a very pessimistic measure: Problems, which are known to be PAC are mostly rather uninteresting, while 'real world problems' (into which category recommender systems surely belong) most likely will *not* be PAC. Its advantage is, that the formalization can be closely related to user modeling processes. It remains to be shown, whether our view on learning user modeling will be a pessimistic one, too.

2.2 What makes a program learn? — Two Examples for Bias

Even though it seems that most interesting problems are not efficiently tractable, one should not just abandon all hope but try to find a subclass of problems which are solvable. The trick is to provide the learning algorithm with extra knowledge and heuristics which guides search and restricts space, that is, *language* and *search bias*. Additionally, one might find a (good) estimate of the unknown distribution. Similarly, the domain sometimes allows for a simplified formalization of the problem which may dramatically decrease computational effort. Finally, it always depends on the desired quality of the sought solution. Quality measures are known as *validation bias*.

However, assumptions made with application of different learning algorithms are known to the expert but they are hidden to the public. Sometimes, this does not matter since algorithms perform well in practice. But in other cases, this may lead to an overestimation of an algorithm's capabilities.

[1] To be precise, ε–good and –bad hypotheses are determined by computing errors on positive examples and negative examples separately, where the error for each set must be less that ε.

2.2.1 *Naïve Bayesian Classifiers.* NBCs (cf. [Cowell et al. 1999]) are very popular in document classification. Given a set of document categories $C = \{c_1, \ldots, c_n\}$ and strings $S = \{s_1, \ldots, s_n\}$, a joint prior probability distribution on observables (that is, preclassified documents) can be approximated by conditional probabilities of string frequencies.[2] A document d is assigned a class c', for which

$$c' = \max P(c|s_1, \ldots, s_n) \tag{2}$$

The above term equals to

$$P(c|s_1, \ldots, s_n) = P(c) \prod_{i=1}^{n} \frac{P(s_i|c, s_1, \ldots, s_{i-1})}{P(s_i|s_1, \ldots, s_{i-1})} \tag{3}$$

which is computationally very expensive due to the chain rule of factorization, see [Castillo et al. 1997]. But by simply *assuming all s_i to be independent of each other* and dropping the constant denominator which does not affect the maximum, one yields a rather cheap function:

$$P(c|s_1, \ldots, s_n) = P(c) \prod_{i=1}^{n} P(s_i|c) \tag{4}$$

This example shows, how simplification of the problem by introduction of a very restrictive language/search bias yields an efficiently solvable problem from a literally intractable problem. Furthermore, naïve Bayesian classifiers deliver fast and precise results which withstand most comparative evaluations. Finally, they are easy to implement for they do not need any background knowledge. Accordingly, they have become a very popular approach for quick document classification, [Billsus and Pazzani 1997; Craven et al. 1998a].[3]

2.2.2 *Induction of Decision Trees.* Decisions trees are concept descriptions which classify objects of the domain by checking the value of most discriminative features. An exhaustive search for the smallest decision tree which is **s**–consistent is NP–complete (shown by Hyafil and Rivest in 1976). Therefore, finding the most informative feature is carried out using an information gain method which is based on information theoretic entropy measures, [Shannon and Weaver 1949].

We assume that all objects $x \in \Omega$ are described by a set of features Φ. The entropy of a set $S \subseteq \Omega$ with respect to a feature $f \in \Phi$ is measured by

$$H(S) = -\sum_{i=1}^{|cod(f)|} p_i \log_2 p_i \tag{5}$$

If, for example, $cod(f) = \{a, b\}$ and $P(f(x) = a) = P(f(x) = b) = 0.5$ for all $x \in \Omega$, then the entropy of any concept with respect to f would be

$$H(S) = -\frac{1}{2} \log_2 \frac{1}{2} - \frac{1}{2} \log_2 \frac{1}{2} = 1 \tag{6}$$

[2]Words, phrases or n–grams.

[3]It is worth a note, that the need for simplification is mentioned in [Craven et al. 1998b] but not in [Craven et al. 1998a].

In order to give an estimate of the quality of a decision node, we need to determine the entropy with respect to $\mathbf{t}$. Accordingly, equation (5) can be rewritten as

$$H(S) = -\sum_{v \in \{1,0\}} \frac{|S \cap \{x : \mathbf{t}(x) = v|}{|S|} \log_2 \frac{|S \cap \{x : \mathbf{t}(x) = v|}{|S|} \tag{7}$$

If entropy of S is considered to be too large for the current validation criterion, S needs to be further divided using another feature f'. Choice of f' is determined by expected information gain: Supposing, that S will be further partitioned using f', expected entropy will be

$$E(S)_{f'} = \sum_{v \in cod(f')} \frac{|\{x \in S : f'(x) = v\}|}{|S|} H(\{x \in S : f'(x) = v\}) \tag{8}$$

which is the sum of all resulting entropies weighted by their relative size. This value is computed for all f' and finally, f with maximum information gain is chosen:

$$f = max(\{f' : f' = H(S) - E(S)_{f'}\}) \tag{9}$$

Basically, information gain based methods have the same domain requirements as naïve Bayes plus a very effective search bias realized by information gain.

First, unconditional probabilities of joint events presuppose a finite set of features with a finite set of feature values each. While the first restriction is not of any effect in applications, the second one might become important when dealing with numeric values. Following the argument on conditional probabilities in naïve Bayesian classifiers, a similar argument applies to information gain as well: In

$$H(X_1, \ldots, X_n) \leq H(X_1) + \cdots + H(X_n) \tag{10}$$

equality is only obtained in cases where X_i are pairwise conditionally independent (see, e.g., [Welsh 1991; Li and Vitanyi 1997]). Accordingly, information gain as defined above overestimates key features (i.e., injective functions). As a consequence, the gain functions have been refined and optimized. For example, gain ratio takes into account entropy with respect to the chosen feature as an additional weight; see [Quinlan 1993] and many others.

2.3 What makes an interface adapt?

An adaptive user interface tries to increase usability by learning about the user and adapting an internal user model. The internal user model determines the behavior of the interface, and it is refined by reasoning about observable user interactions, [Jameson 2003].

Let us assume, that the user model $\mathbf{M}_u$ for a user u shall predict a user's behavior as it is recorded by observing user interactions. Accordingly, a user modeling problem can be defined canonically: A learning algorithm $\mathbf{A}$ induces a user model $\mathbf{M}_u$ (that is, a hypothesis h) based on background knowledge Σ and feedback $\mathbf{f}$ (a sample $\mathbf{s}$). Feedback is recorded by collecting and *interpreting observable* interactions:

$$\mathbf{f} = F(m, \mathbf{o}) = [\langle x_1, \mathbf{o}(x_1)\rangle, \langle x_2, \mathbf{o}(x_2)\rangle, \ldots, \langle x_m, \mathbf{o}(x_m)\rangle] \tag{11}$$

The user's interest $\mathbf{i}_u$ makes the user perform certain actions which can be observed, but the user's intention for performing this action remains unknown to the system.

Accordingly, the usage of $\mathbf{o}$ as a label in the sample means that the learning algorithm will not immediately try to predict $\mathbf{i}_u$ (that is, $\mathbf{t}$), but it will try to predict what the next interaction will be. Figure 2 shows the parallels of machine learning (lower part) and user modeling (upper part).

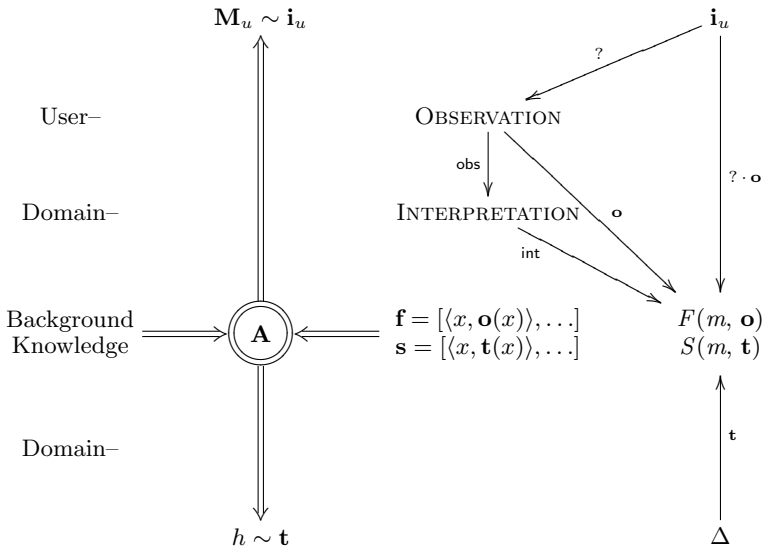


Fig. 2. Parallels and differences in ML and UM

At this point, it becomes clear, that the behavior of F is unclear at two different levels:

(1) The intention behind performing $\mathbf{o}(x)$ is veiled by the fact, that the user's tries to realize his interest $\mathbf{i}_u$ based on his very own idea of the functionality of interactive elements as provided by the system. This is represented by the '?'–arrows in in figure 2.

(2) The label $\mathbf{o}(x)$ that is taken as a teacher signal for the learning task is based on the system's built–in interpretation of the user's interactions. As such, it is subject to noise for two reasons:
First, the observation itself may be vague or noisy (obs). Second, the interpretation of the user's interactions is subject to domain simplification as well; but it is also prone to generating a noisy signal (int).

Concluding, it seems that a closer investigation on how interactions pertain to user interests and feedback labels illucidated the problems one encounters when trying to quickly induce user models from observation. Trying to summarize our considerations in one sentence, a pessimist might conclude that:

> Machine Learning in User modeling tries to mimmick a user's behavior,
> but it does not model a user.

Accordingly, an adaptive recommender system will be able to recommend by predicting most probable actions—but it might not be able to *explain* why this recommendation was made in a way that is understandable to the user. Even though this violates the strong definition of Machine Learning, such a learning adaptive system is capable of dealing with vast amounts of data and few backround knowledge such that it is very popular in collaborative filtering.

3. FINDING OUT ABOUT THE UNKNOWN: UNDERSTANDING F

As introduced in the last section we will investigate the nature of feedback $F(m, \mathbf{o}) = \mathbf{f}$ by further decomposing F and $\mathbf{o}$. For the remainder of the article, $i'_u(x)$ will denote the *intended* (that is, the *planned*) action on x by u which is motivated by the user's interest $\mathbf{i}_u$ and his understanding of the system. $\mathsf{obs}_u(x)$ is the observed interaction by the system which is interpreted by int. Accordingly, the following relations hold in general:

$$\mathbf{i}_u(x) \neq i'_u(x) \tag{12}$$

$$\mathbf{o}(x) = \mathsf{int}(\mathsf{obs}(x)) \tag{13}$$

$$\mathbf{o}(x) \neq \mathbf{i}_u(x) \tag{14}$$

An very well–known example is collaborative filtering as it is performed by, e.g., Amazon. In collaborative filtering or user modeling, the 'user model' consists of items or objects of the domain which are associated to the user. Similar sets of entities describe similar users. Accordingly, observed interactions ($\mathsf{obs}(x)$) are purchases of items—but the reason behind a specific purchase is unknown. Here, the sample that is taken as input, $\mathbf{o}(x)$, is assumed to be a sign of interestingness to the user by way of $\mathsf{int}(\mathsf{obs}(x))$.

There are, however, systems which try to model $\mathbf{i}_u(x)$. Others try to reason about user's plans and thus try to predict $i'_u(x)$. Further systems take into account cognitive states as a part of a user model. We will now investigate a few recommender systems using this framework in order to obtain a clear picture on working systems which learn user models.[4]

3.1 Naïve Bayes in UM Applications

As already stated earlier in this paper, NBCs are very popular. We will just briefly describe two different projects which illucidate the wide variety of possible application areas and implications.

3.1.1 MANIC*: NBCs learn about learners.* In relation to a student's knowledge level and preferences for presentation, the MANIC system tries to learn, whether a student would '*want*' to read a certain paragraph within an instructional text, and thus, whether it is worth a recommendation. Accordingly, the objects of Ω are information objects (including multi-medial objects such as graphics or animations) which are linked by relations to form a semantic web which describes dependencies between lectures topics.

[4]The list of discussed approaches is by no means complete or even remotely exhaustive. I simply chose several more or less well known projects which allow to discuss this paper's issues from a wide diversity of perspectives.

A lecture 'slide' is generated by choosing a topic and then traversing the network, thereby collecting all information objects needed for the presentation. For each optional node, it has to be decided, whether the student might want to see it (i.e. whether a swith for unfolding text should be displayed). The features used to describe these items are *media type*, *instructional type*, *abstractness*, place in *topic* and *concept* and finally the target feature *wanted*.

A NBC is used to learn and then predict whether a student might *want* to see a certain item. Observable actions are clicks on offered stretch-out icons: If the student unfolds an offered node, this is taken as positive feedback; not stretching a text of his current level of expertise is interpreted as negative feedback.

The overall result of this approach is that a student will learn better because he will more like the presented material as content and presentation quickly adapt to his desires.

It is clear, that in this case, $i'_u(x)$ and $\mathsf{obs}(x)$ will match pretty well. However, the problem there remains the problem of interpreting 'not clicking' as 'not wanting': For an object x of matching difficulty, $\mathbf{o}(x) = 0$ implies $\mathbf{i}_u(x) = 0$ which is not necessarily true. Similarly, it has to be evaluated, whether the features for describing objects of the domain are independent.

Nevertheless, a domain restriction, domain simplification, and few interactive elements allow for using the chosen ML method and receiving promising results.

The MANIC system was presented in [Stern and Woolf 2000].

3.1.2 *Personalized* IR *with* NBC*s.* The METIORE system, [Bueno and David 2001] also uses NBCs for recommendation of articles. In this paper, it is clearly stated, that:

> *[...] the result of the algorithm will be the degree of relevance of an object to the present objective for a user.*

The interesting thing about it is, that feedback is observed not only as a binary event but recorded as *relevant*, *relevant but already known*, *indifferent* and *irrelevant*. Similar distinctions are made within the NEWSDUDE, where also NBCs are used for recommendation of news items, [Billsus and Pazzani 1999]. The NEWSDUDE focuses especially on the difference between long–term and short–term interests: '*... a user's information need changes as a direct result of interaction with information*'.[5] The system reads or displays news to the user and awaits explicit feedback, including two further types of positive feedback ('interesting, but I already know', formerly often covered by negative feedback, and 'tell me more!'). Accordingly, here the same considerations apply as for the above mentioned MANIC system.

However, objections against NBC approaches in this domain have been discussed in [Billsus and Pazzani 1997]. SYSKILL & WEBERT is a meta search engine which offers the opportunity to give explicit feedback for each result. The feedback is used to build an individual user model that contains sets of boolean key word vectors. Using trained Bayesian classifiers, web documents (i.e. links on currently

[5]It is worth a note, that the NEWSDUDE technology has now been further developed into a commercial product (ADAPTIVE INFORMATION SERVER, cf [Billsus et al. 2002]) with many features similar to the academic INFORMATIONVALET project, [Macskassy et al. 2000].

displayed pages) are recommended with respect to the user model. In their article, the authors show that the violation of the independence assumption does not affect NBCs to '*perform well [...] in many domains that contain clear attribute dependence*'. However, the NBC approach is refined by the authors: Since NBCs try to approximate joint probability distributions based upon observations, the prior probabilities when learning 'from scratch' may have crucial effect on the overall accuracy (initial 'near zero probabilities' may lead to a myopic feature deletion). Accordingly, additional assumptions (ε–probabilities and Laplace distribution) on the prior distribution are made—and these assumptions are made with respect to the class that is to be described:

> *[...] The reason, why a strong estimation bias is beneficial in some domains, but hurts performance in others, seems to be linked to the "quality" of extracted features.*

As a consequence, the author's algorithm chooses one of the aforementioned assumptions based on an information gain estimate of a feature's quality. Again, one might criticize the use of information gain methods—but again, results show that this method and it's implicit requirements (though violated) work well on the domain.

3.1.3 *Summary:* NBC*s in* AUI*s.* One should keep in mind, that the target learned by NBCs is $\mathbf{o}(x)$ and thus models the user behavior as observed and interpreted by the system, but not the user's interest itself. Furthermore, the application of NBCs presupposes severe domain restrictions and biases based on assumptions that in general do not hold.

On the other hand, it has been shown, that NBCs perform well though; which is partially due to the 'simple' task of predicting few observable interactions in restricted domains. Therefore, a bigger set of observable user interactions (like $cod(\mathsf{int}) = \{$ *interesting*, *interesting-but-already-known*, *uninteresting*, $\ldots\}$) and a pretty straightforward, and thus noise free, interpretation thereof should be seen as a major improvement when trying to achieve equality in equations (12) and (14).

[Pretschner and Gauch 1999; Madrid and Gauch 2002] and [Müller 2002] give a more detailed description of ongoing research in related fields.

3.2 Causal Models & Bayesian Networks

It seems obvious, that knowing about a user's cognition and emotion helps to explain the difference between intention i'_u and interest $\mathbf{i}_u(x)$, and thus may help to make understand observed data $\mathbf{o}$ as feedback $\mathbf{f}$. Furthermore, recent advances in multi modal interfaces supplies us with much more data of different quality: pulse rate, breath rate, speech stress patterns, galvanic skin response. Such data can be used to build hypotheses about a user's emotional or cognitive state. When working with data from multiple resources, interpretation and integration is not trivial. Again, machine learning seems to offer a simple solution: Feed data into the system and let the system learn to interpret and integrate data.[6] Again, observations

[6] A prototypical example for this modus operandi was the hype followed by the resurrection of Artificial Neural Networks: Feed any data into the input layer of a Multi–Layer

can be predicted, but they can not be explained by decomposition in intention and realization. As a consequence, the user model may take into account further data but it still does not explain a user's needs or cognitive or emotional needs. Therefore, more sophisticated models are needed.

3.2.1 *Probabilistic estimates of a user's cognitive states: Bayesian Networks.* High cognitive load of a user can, for example, be detected by analysis of speech (silent and filled gaps, syllable frequency).[7] General preferences (and, for example, personality) and cognitive load have an impact on a user's behavior.

Consider the following problem: Observing a CANCEL event, we want to know, whether this is due to disinterest d in a displayed item or impatience i while d and i are assumed to be independent. Given prior probabilities, Bayes' theorem allows for interpreting causal relationships by a prior–to–post and a post–to–prior decomposition. We represent the events by variables C, D, and I and abbreviate $P(C = \texttt{yes})$ by $P(c)$ (and the reverse case by $P(\bar{c})$). Supposing, that the prior probabilities of a web page being not interesting d is 0.5, and the chance for a user being impatient i is 0.2, one can derive a joint probability distribution. Both D and I are possible causes for c, and the likelihood of observing c given D and I can be inferred from previously observed events. We assume the following probabilities as shown in the leftmost column of table I. By multiplying $P(I)$ and $P(D)$ one

Obervables		Priors	Posterior	Abduction
D	I	$P(c\|D,I)$	$P(D,I\|c)$	$P(D,I\|c,a)$
d	i	0.5	0.33	0.49
d	$\bar{i}$	0.8	0.33	0.07
$\bar{d}$	i	0.7	0.28	0.41
$\bar{d}$	$\bar{i}$	0.1	0.07	0.02

Table I. Bayesian Resoning

receives the joint probability distribution $P(I, D)$ (because I and D are assumed to be independent). Taking into account conditional probabilities from above, Bayes' law can be applied to calculate $P(I, D, C)$ and, finally, the posterior probabilities $P(D, I|c)$. However, this insight is not really helpful: Both d and i are nearly equally good explanations for c.

Instead, the Bayesian network is extended by a further variable, which depends only on one of the two observable events: For this example, we assume that being angry A depends only on the personality of the user. Accordingly, we need a new local distribution which explains the probability of being angry given an impatient personality. Here, we choose

$$P(a|D, C, i) = 0.6 \text{ and } P(a|D, C, \bar{i}) = 0.1$$

Perceptron and let the hidden layers learn to map input patterns to output patterns.

[7]There are according 'symptoms' in the context of interactions with graphical user interfaces, too: [Lindmark 2000] gives a broad overview of observable events together with their interpretation as symptoms for resource limitations. The problem of learning to recognize emotions from speech is discussed in [André and Müller 2003].

Now, the posterior distribution $P(D, I, c)$ is taken as new prior distribution to calculate $P(a, D, I|c)$ and finally $P(D, I|a, c)$.The overall outcome of this calculation is, that:

> If we observe c and we know that the user is angry (a), then:
>
> —disinterest d is be the reason in 56% of all cases, while
> —the probability of having an impatient (i) user is 0.90

In such a case, this outcome would be a strong argument for not labeling the document being presented to the user as uninteresting—because the negative feedback of the user is much more likely caused by his impatience.

Bayesian networks are a very popular approach in user adaptive interfaces, because they inherently provide us with a very elaborate theory on how to deal with vagueness. However, with growing networks, computational effort increases more and more. However, one question remains: Where do prior probabilities and the topology of the network come from? Here, substantial work has been carried out in course of the READY project. The READY system tries to recognize, reason about and adapt to a user's cognitive capacity which is, e.g., determined by time pressure, lack of knowledge or cognitive load. Two scenarios are a telediagnosis for a car breakdown, where the user is stressed by the problem, noise, and time pressure and a resource adaptive travel assistant for the Frankfurt airport.

The structure of an underlying Bayesian network can be justified by a causal network whose architecture and initial probabilities have been empirically validated, [Jameson et al. 2000] (see also footnote 7). In this system, the user model $\mathbf{M}_u$ shall not primarily describe a user's interest, but his cognitive state. As mentioned above, the interpretation of observed symptoms is empirically validated, such that the noise caused by $\mathsf{int}(\mathsf{obs}(x))$ can be assumed to be minimal. Accordingly, this approach can be regarded to an approach where $\mathbf{f}$ delivers examples of very high quality. On the one hand, the Bayesian network can be used to predict user behavior (lack of knowledge increases the probability of 'description errors' and thus, 'click and re-click'–events) but also to explain observed events ('overscrolling' is an observation that supports the symptom 'motor error' which could be explained by time pressure).[8]

3.2.2 *I think you know I think: A user's idea of a system's user model.* For describing interaction scenarios like dialogs, mutual models need to taken into account. When asking someone for a favor, our model of the dialog partner is crucial for planning our speech act in a way that it determines whether to ask at all, and then, how to ask. Accordingly, early work on user modeling was based on belief models in dialogs, [Kobsa and Trost 1984].

It is clear, that a user has a model about the system he is working with and so adaptive systems might try to learn a model about the user. In agent communities

[8]Examples are chosen in concordance to figure 2 in [Lindmark and Heckmann 2000]. Similar examples with spoken language as observed interactions can be made up from the BN shown in figure 3 in [Müller et al. 2001]. The latter article also describes how to design and validate the architecture of a BN based on an empirical experiment.

(an abstraction from human–computer interaction), several individuals collaborate and to achieve a common. [Suryadi and Gmytrasiewicz 1999] describe such a setting for goal directed collaboration based on assumptions on each other's capabilities and needs.

Knowing about a user's model of the system may help to understand the difference between $\mathbf{i}_u(x)$ (his needs) and $i'_u(x)$ (his plans by which he tries to achieve his goal). One the one hand, we want to learn some model $\mathbf{M}_u$ which shall approximate but on the other hand, $\mathbf{f}$ is based on $\mathbf{o}(x)$, and thus on $i'_u(x)$ and not $\mathbf{i}_u(x)$.

A very primitive, implicit mutual model is used in systems, where the user's acquaintance or skill level is described by attributes such as *beginner*, *intermediate* or *expert*, which, on the other hand is not very reliable as well.

3.3 Learning to explain Recommendations

When trying to learn a user's plan one needs to be able to describe a plan. Similarly, complex domain descriptions with lots of background knowledge involved need a richer description language. Sometimes, a user model rather consists of 'programs' describing procedural behavior which are able to predict or recommend actions or items. Furthermore, the need for scrutable user models requires a system to describe its behavior to the user.

One machine learning technique which is able to satisfy these needs is the underestimated approach to learning user models by inductive logic programming (ILP).[9]

However, there are few recent approaches in UM which make use of ILP because of its abovementioned advantages.

3.3.1 *Learning to filter e-mail.* [Kay and McCreath 2001] describe an ILP approach to the problem of personalized mail filtering, MUMILP. Mail filtering is desperately needed in recent times and therefore has become a popular application area in user modeling. Furthermore, learning how to filter mail is supported by clearly labeled examples: Moving mail into folders delivers reliable classification data as a teacher signal $\mathbf{t}$. Learning mail filters seems not to be a recommendation task at first blush, but as we shall see, a theory of mail filters can be used to predict a filing task and, therefore, for recommending how to classify mails.

The classification task is twofold: Filtering spam is a binary decision, while preordering of mails into existing folders is a more sophisticated learning task. MUMILP learns rules which preclassify incoming mails. In terms of ILP or, more general, relational learning, the task is to learn a relation between messages and folders—which is performed by relating strings describing the folder's name to strings from the mail.

Here, the domain model requires e-mails to be represented in a way the ILP–algorithm can construct hypotheses. Accordingly, the predicates used determine the sender, occurrence of a string etc. Example hypotheses could be:[10]

[9] It is an interesting fact that ILP has not been considered as a promising technique in ML4UM in the past years, especially since the first modeling approaches to user modeling were logic based (and still, some are as we have seen in the section on programming by demonstration); see, e.g. [Kobsa 1988].

[10] Slightly changed from [Kay and McCreath 2001].

```
filter(M, spam) :-
   mail_subject(M, X), member(cash, X).

filter(M, friends) :-
   sender_identity(M, X), member(X, [jack, jill]).

filter(M, mllist) :-
   messages(X, mllist), most_similar(X,M).
```

The first one tries to identify spam mail by the key word `cash` in the mail subject. The second one classifies mails from `jack` or `jill` as mails from friends. Finally, the third rule hands the problem of text classification over to some TfIdf classifier by comparing the incoming mail with all folders and succeeding, if `mllist` is the folder that yields the best similarity value. Of course, all those literals could be combined to gain more complex rules.

In this approach, **f** can be assumed to be rather noise–free; as long as we want to discriminate spam from non–spam. Learning more sophisticated classifiers is more prone to noise as a single mail may belong to several classes. On the other hand, the fact that a certain mail belongs to folder `recreation` supports the proposition that it does not belong to the folder `conferences`. But if we are concerned with folders for several conferences with overlapping committees this discrimination is not as easy as in the first case (this problem has been worked in the approach described in the next paragraph). However, one can agree that a user files a mail with purpose (in order to retrieve it quickly) and thus, the intention behind filing is the same to all users. Similarly, the events that are being observed are easy to interpret: Filing a mail is a clear positive evidence for a classifier.

In this approach text clues are taken into account when trying to induce user models. Another idea is to discriminate text classification from user model learning. This has been done in the project described in the next paragraph.

3.3.2 *Learning about interesting documents in web search.* Another project, which we want to describe only very briefly is OySTER, where Ilp was used to learn user models which contain logic programs describing a user's interest, [Müller 2001]. Documents on the web were classified into a taxonomy of content–specific concepts and document type specific information. The domain description language of concepts and types is used for both the documents and the user models. Upon submitting a search request to a meta search engine, OySTER collects all documents which then can be compared to the user model. The data taken as input consists of the documents together with a label that is derived from explicit feedback. The user model contains several aspects of a user's interest where each aspect consists of a set of clauses representing a procedural description of 'interestingness'.[11] One such clause could, for example, state, that a document is interesting, if: it is a 'scientific publication', it deals with 'user modeling', and it belongs to the concept 'machine learning' with a confidence of at least, say, 75 per cent. The

[11]This method can be vaguely related to the Foil learning algorithm, where rules become more accurate by adding body literals as constraints and the whole program becomes more general by inventing more rules.

use of different aspects allows for a more precise (that is, accurate) description of different topics of interest.

The primary aim behind this approach was to overcome the lack of feedback: with only a very small sample $\mathbf{f}$, a learning algorithm will produce a rather bad hypothesis (which will be either too general or too specific). By taking positive evidence for one aspect as negative evidence for another aspect, the sample could be enlarged such that a better user model could be learned. Of course, this method cannot work in scenarios where sufficient feedback for learning is already available: The more feedback we have, the bigger the danger of erroneously counting a 'foreign' evidence as a positive or negative for interest—especially in context of overlapping interest aspects or documents which may belong to several classes. Speaking of classes, another disadvantage of this approach becomes clear: One taxonomy for all presupposes an equal understanding for all users which certainly is not the case.

However, it was shown, that for few observed events with a clear interpretation (obs records explicit feedback which describes interestingness of documents as int) accuracy could be significantly improved while for a sample size $m > 25$ the accuracy gain vanished; see [Müller 2002]. Another big advantage of this approach is that the hypotheses is a user model $\mathbf{M}_u$ is both a scrutable description of a user's interest as well as a procedure for user specific recommendation.

As one can see, relational learning allows to induce complex descriptions which go beyond what simple classifiers can deliver. It can deal with much more complex domains including background knowledge. The disadvantage is, however, that without a decent domain theory or no background knowledge, one cannot expect impressive results and thus, some effort of domain modeling is required. This might be the reason for why Bayesian approaches now play such a dominating role in recommender systems and adaptive systems in general.

4. IS MOST PROBABLE BETTER THAN GOOD?

It has been shown, that many different approaches allow to predict a user's behavior and thus, to recommend further actions (the discussion in this article could be extended to artificial neural networks and evolutionary algorithms). Some of these approaches are rather data driven, some are (much) more sophisticated and try to induce a real model which explains what is going on in a user's mind.

It has to be emphasized though, that learning is not an all–purpose tool for adaptive systems: Sometimes it takes too long, sometimes there is too few data, sometimes there is too few knowledge available, and sometimes, the tool simply does not fit the problem. Concluding, we will just point out a few issues that one might like to consider when thinking about which learning algorithm to apply within a user adaptive recommender system.

4.1 Good, better or most probable?

A constructive and still very cautious approach to adaptive systems in general has been presented in [Miller 2000]. It is argued, that Gricean conversational maxims, [Grice 1975], also apply to human–computer interaction. Any 'good' adaptive user interface must provide the user with the right information and it must provide the

user with the right amount of information and it must present this information in the right way to the user.

In presence of huge data sets, where we quickly need a probably right piece of information, and do not need to take care of other conversational maxims, simple Bayesian Classifiers seem to be the best choice (even if the requirement of independence might be violated).

But as it is essential for the user to have the feeling of understanding the system, the user needs to have an idea about what the systems knows and what the system assumes about the user. In this way, the user is able to understand *why* the system performed a certain step towards adaption. This need favors algorithms like Bayesian Networks or relational learning wich require more or less beckground knowledge and examples. While domain knowledge costs effort of a knowledge engineer, feedback is required from the user.

But as we try to decrease cognitive a user's cognitive effort, adaptive systems need to act in the background. On the other hand, user adaption without feedback is impossible. This seems to conflict with the moist important conversational rule in HCI: *Never ever bother the user.*

As a consequence, a good recommender system should be able to interpret the *natural* user behavior on all levels as user feedback—without the need for any further explicit feedback. The user model built upon this knowledge shall be easy to understand for the user, the adaptive process itself should be scrutable, self–explaining and failsafe. At this point the dilemma becoms clear: Being able to work with few or vague feedback means to be unable to be scrutable, asking for feedback means to bother the user and interpreting the user's natural interaction will increase noise in $\mathbf{f}$.

4.2 Probably approximately correct recommendations?

In section 2.2 we have described the PAC learning model. Throughout this article, the user modeling problem was related to the machine learning problem. Accordingly, one would have to determine, whether some $\mathbf{M}_u$ is 'ε–good'. In cases, where Δ is unknown, hypotheses h are evaluated against a test set $\mathbf{f}_{\text{test}}$ which is taken from the sample $\mathbf{f}$.[12] Assuming that the the sample somehow represents Δ on Ω (which is the underlying inductive hypothesis), the error set on $\mathbf{f}_{\text{test}}$ can be used to estimate ε. However, the method of testing with $\mathbf{f}_{\text{test}}$ cannot be compared to machine learning: The aim in machine learning is to satisfy $h \sim \mathbf{t}$ (which can be tested by the method described above), but in user modeling, the sample $\mathbf{f}$ is used to learn some $\mathbf{M}_u \sim \mathbf{i}_u \neq \mathbf{o}$. Accordingly, the error measured when comparing $\mathbf{M}_u$ and $\mathbf{o}$ is the predictive error of $\mathbf{M}_u$ on observable user interactions—not the error in the user model itself.

To satisfy the requirements of HCI, one would need to test the agreement between $\mathbf{M}_u$ and $\mathbf{i}_u$. For this, one would need either an expert who evaluates $\mathbf{M}_u$ or an experiment where we would have to measure accuracy in interactions with real users.

Finally, determining a confidence in ε–goodness of hypotheses one needs to consider the probability of being able to learn a user model which has an error of at

[12] As in machine learning, examples from $\mathbf{s}_{\text{test}}$ ($\mathbf{f}_{\text{test}}$) are not taken for learning.

most ε. In the PAC model, this confidence value corresponds to δ. In practice, one would need to verify a user adaptive system in a test series with many participants.

4.3 A good recommendation in the hand is worth a probably better in the bush

The above considerations may seem to be pure sophistic, but we always should be concerned to be clear on what is being modeled, what the domain restrictions look like and what biases are involved.

Accordingly, one would always try to find the *best* solution for a certain recommendation, and depending on these requirement, it might be the good, or the better or the most probable one. A recommender system can be learned, if as many as possible of the following restrictions can be satisfied:

(1) The content of the user model is self–contained and clearly restricted as well as the domain
(2) If the domain is sufficiently well described
(3) If there is a legitimate trade–off between scrutability, adaptivity and domain modeling effort
(4) If we can live with assumptions on the domain and biases in learning
(5) If there is enough noise–free data available

There is, however, a chance to go a step further: If we are able to achieve more knowledge about the process that makes a user interact in a certain way, we might be able to de–noisify $\mathbf{f}$ by understanding the intended actions. This idea has already been partially worked on in plan recognition and affective interaction, but it seems to be a fruitful further research direction to integrate these approaches. As a consequence, an intelligent, adaptive user interface will be able to 'understand' observed interactions $\mathbf{o}(x)$ and thus yield a better approximation of the teaching signal $\mathbf{t}$.

Scrutable and relational hypotheses and their according learning algorithms will always somehow stand in opposition to fast, 'cheap' and robust recommendations. However, a recent e-mail discussion in the collaborative filtering community has shown, that there is still much work to do in the area of recommender systems. It could be, that further dimension on evaluating the utility of machine learning paradigms in recommender systems will be taken into account pretty soon.

REFERENCES

André, E. and Müller, M. E. 2003. Learning affective behavior. In *Universal Access in HCI. Proc. 10th Intl. Conf. on HCI*, C. Stephanidis, Ed. Vol. 4. Lawrence Erlbaum Associates.

Bauer, M., Gmytrasiewicz, P., and Vassileva, J., Eds. 2001. *User Modeling: Proceedings of the 8th International Conference, UM–2001.* Springer.

Billsus, D., Brunk, C. A., Evans, C., Gladish, B., and Pazzani, M. 2002. Adaptive interfaces for ubiquitous web access. *Comm. ACM 45,* 5, 34–38.

Billsus, D. and Pazzani, M. 1997. Learning probabilistic user models. In *User Modeling: Sixth International Conference, UM97*, A. Jameson, C. Paris, and C. Tasso, Eds. Springer Wien New York, Vienna, New York. Workshop Paper.

Billsus, D. and Pazzani, M. 1999. A hybrid user model for news story classification. See Kay [1999], 99–108.

Bueno, D. and David, A. A. 2001. Metiore: A personalized information retrieval system. See Bauer et al. [2001].

Castillo, E., Gutierrez, J. M., and Hadi, A. S. 1997. *Expert Systems and Probabilistic Network Models*. Springer.

Cowell, R. G., Dawid, A. P., Lauritzen, S. L., and Spiegelhalter, D. J. 1999. *Probabilistic Networks and Expert Systems*. Springer.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. 1998a. Learning to extract symbolic knowledge from the world wide web. In *Proc. of 15th Nat. Conf. on Artificial Intelligence (AAAI-98)*. AAAI Press.

Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. 1998b. Learning to extract symbolic knowledge from the world wide web. Tech. Rep. CMU-CS-98-122, School of Computer Science, Carnegie Mellonc University. 9. Short version published in [Craven et al. 1998a].

Grice, H. P. 1975. Logic and conversation. In *Syntax and Semantics, 3: Speech Acts*, P. Cole and J. L. Morgan, Eds. Academic Press, New York.

Jameson, A. 2003. *Handbook of human–computer interaction in interactive systems*. Erlbaum, Chapter Adaptive Interfaces and Agents.

Jameson, A., Gromann-Hutter, B., March, L., and Rummer, R. 2000. Creating an empirical basis for adaptation decisions. In *IUI2000: International Conference on Intelligent User Interfaces*, H. Lieberman, Ed. ACM.

Kay, J., Ed. 1999. *User Modeling: Proceedings of the Seventh International Conference, UM99*. Springer.

Kay, J. and McCreath, E. 2001. Automatic induction of rules for e-mail classification. In *Proceedings of the UM2001 Workshop on Machine Learning for User Modeling*, R. Schäfer, M. E. Müller, and S. A. Macskassy, Eds. 59–66.

Kearns, M. J. 1990. *The Computational Complexity of Machine Learning*. MIT Press.

Kearns, M. J. and Vazirani, U. V. 1994. *An Introduction into Computational Learning Theory*. MIT Press.

Kobsa, A. 1988. User models in dialog systems. In *User Models in Dialog Systems*, A. Kobsa and W. Wahlster, Eds. Springer.

Kobsa, A. and Trost, H. 1984. Representing belief models in semantic networks. In *Cybernetics and Systems Research II*, R. Trappl, Ed. North–Holland.

Li, M. and Vitanyi, P. 1997. *An Introduction to Kolmogorov Complexity and its Applications*, 2nd ed. Springer.

Lindmark, K. 2000. Interpreting symptoms of cognitive load and time pressure in manual input. M.S. thesis, Department of Computer Science, Saarland University, Germany.

Lindmark, K. and Heckmann, D. 2000. Interpreting symptoms of cognitive load and time pressure in manual input. In *Proc. 8th GI Workshop "Adaptivität und Benutzermodellierung in interaktiven Softwaresystemen"*, M. E. Müller, Ed. Institut für Semantische Informationsverarbeitung, Universität Osnabrück.

Macskassy, S., Dayanik, A. A., and Hirsh, H. 2000. Information valets for intelligent information access. In *Adaptive User Interfaces*, S. Rogers and W. Iba, Eds. Number SS–00–01 in Technical Report. AAAI Press, Menlo Park, California, 68–73.

Madrid, J. M. and Gauch, S. 2002. Incorporating conceptual matching in search. In *11th Conference on Information and Knowledge Management (CIKM'02)*. Submitted.

Miller, C. A. 2000. Rules of Etiquette — or How a Mannerly AUI should Comport Itself to Gain Social Acceptance and be Perceived as Gracious and Well-Behaved in Polite

Society. In *AAAI Spring Symposium on Adaptive User Interfaces*. AAAI Press, Menlo Park, 80–81.

Müller, C., Grossmann-Hutter, B., Jameson, A., Rummer, R., and Wittig, F. 2001. Rcognizing time pressure and cognitive load on the basis of speech: An experimental study. See Bauer et al. [2001].

Müller, M. E. 2001. Inducing content based user models with inductive logic programming techniques. In *Proceedings of the UM2001 Workshop on Machine Learning for User Modeling*, R. Schäfer, M. E. Müller, and S. A. Macskassy, Eds. 67–76.

Müller, M. E. 2002. Inducing conceptual user models. Ph.D. thesis, Institute of Cognitive Science, University of Osnabrück. Electronic publication.

Müller, M. E. 2004a. *Adaptable and Adaptive Hypermedia Systems*. Idea Group, Chapter Learning adaptive Behavior. To appear.

Müller, M. E. 2004b. Can user models be learned at all? *Knowledge Engineering Review xxx,* xxx. In press.

Pretschner, A. and Gauch, S. 1999. Personalization on the web. Tech. Rep. ITTC–FY2000–TR–13591–01, Dep. Electrical Engineering and Computer Science, Univ. Kansas.

Quinlan, J. R. 1993. *C 4.5 — Programs for Machine Learning*. Morgan Kaufmann.

Shannon, C. and Weaver, W. 1949. The mathematical theory of communication. Tech. rep., University of Illinois, Urbana.

Stern, M. K. and Woolf, B. P. 2000. Adaptive content in an online lecture system. In *Adaptive Hypermedia and Adaptive Web–Based Systems: AH-2000*, P. Brusilovsky, O. Stock, and C. Strappavara, Eds. Number 1892 in LNCS. Springer.

Suryadi, D. and Gmytrasiewicz, P. 1999. Learning models of other agents usinginfluence diagrams. See Kay [1999].

Valiant, L. G. 1984. A theory of the learnable. *Communications of the ACM 27*, 1134–1142.

van Rijsbergen, C. J. 1979. *Information Retrieval*. Butterworths.

Webb, G. I., Pazzani, M. J., and Billsus, D. 2001. Machine learning for user modeling. *User Modeling and User–Adapted Interaction* 11, 19–29.

Welsh, D. 1991. *Codes and Cryptography*. Clarendon – Oxford University Press.